MCSim: A Discrete-Event Simulation Framework for Studying Multi-Cloud Networks

Joseph Colton
University of Oregon

Ramakrishnan Durairajan
University of Oregon and Link Oregon

***Abstract*—As enterprises adopt multi-cloud strategies understanding network behavior across heterogeneous cloud infrastructures becomes critical for performance and cost optimization. However, existing network simulators lack support for cloud-native hierarchies (providers, regions, zones) and cannot incorporate empirical inter-cloud measurements or egress cost modeling. We present *MCSim*, a discrete-event simulator designed specifically for empirical trace-driven multi-cloud network analysis. *MCSim* models hierarchical cloud topologies with time-varying metrics derived from real measurements, supports cost-aware routing policies, and also supports hybrid emulation by applying simulated network conditions to real hosts using Linux traffic control (tc), bridging simulation with real-world validation. We validate *MCSim* using 50+ million measurements across 105 VMs in AWS, GCP, and Azure regions. Statistical analysis shows *MCSim* achieves $<$1% error in average latency reproduction, with 87% of inter-cloud pairs showing $<$5ms RMSE. Case studies demonstrate *MCSim*'s ability to analyze cost-performance trade-offs (27% latency reduction with 57% cost increase), simulate failure scenarios (i.e., impact from regional outages), and enable hybrid testing with $<$2% deviation from real deployments. *MCSim* scales to 1000+ nodes with spatial parallelization, enabling reproducible multi-cloud experimentation at unprecedented scale and fidelity.**

***Index Terms*—Multi-cloud networks, simulation framework**

I. Introduction

Modern enterprises are increasingly adopting multi-cloud strategies to enhance resilience, optimize costs, and avoid vendor lock-in across diverse domains such as genomics, healthcare, finance, among others [1]. This shift enables enterprise applications to span multiple cloud providers, leveraging the strengths of each platform. However, operating in a multi-cloud environment introduces complex challenges, especially under dynamic conditions such as variable network latency, fluctuating egress costs, and outages. For example, fintech companies such as Form3 [2] and Stake [3], deploy latency-sensitive applications across different providers (e.g., AWS, GCP, Azure, etc.) to serve global markets while complying with regional regulations [4]. During peak trading hours, these systems must navigate unpredictable inter-cloud latencies, shifting cost structures, and complex failover scenarios. Understanding the behavior of such deployments, especially how routing decisions impact both cost and performance during disruptions, is crucial for ensuring reliable, efficient operation. However, evaluating such interactions in production is costly and risks disrupting services [5].

Existing network simulation tools fall short of multi-cloud requirements in three critical ways. First, traditional simulators like NS-3 [6] and OMNeT++ [7] focus on protocol-level behavior but lack abstractions for cloud-native hierarchies and cannot model cloud-specific pricing like egress costs. Second, cloud-specific simulators such as CloudSim [8] model resource allocation but abstract away network details, ignoring real-world routing dynamics and inter-cloud latency variability. Third, no existing tool can incorporate empirical measurements from live multi-cloud deployments, forcing operators and researchers alike to rely on synthetic parameters that poorly represent actual cloud behavior (see Table I).

In response, we present *MCSim*[1], a discrete-event high-level packet simulator designed specifically for multi-cloud network experimentation. Built atop the lightweight SimPy engine, *MCSim* allows users to construct and explore cloud-scale environments spanning providers, regions, and zones, with topologies defined at a fine-grained level. Links between nodes represent real-world metrics such as latency, jitter, packet loss, and egress cost, which can either be user-defined or replayed from empirical measurement data. Crucially, all network parameters in *MCSim* can be static (in the form of configuration files) or dynamic (i.e., from live multi-cloud measurements), enabling dynamic experimentation under changing network conditions, failures, or cost models.

MCSim incorporates a pluggable routing engine that supports shortest-path, overlay, and policy-based routing strategies, optimized for latency, cost, or reliability. It allows researchers and practitioners to study how network behavior evolves in response to real-world dynamism such as how traffic reroutes during regional failures, or how cost-efficient paths emerge under bandwidth constraints. Users can also simulate common application behaviors (e.g., ping, traceroute, TCP/UDP flows), define custom traffic generators, or experiment with new protocol designs.

MCSim addresses the scalability challenges of routing across different cloud providers to mitigate performance issues while being optimal w.r.t. diverse egress costs and varying network conditions through two key innovations: caching and spatial parallelization. The system implements route caching to store previously computed paths with their metrics (latency, egress costs) and snapshot caching to capture network states at intervals, both eliminating redundant computations. Additionally, *MCSim* uses spatial domain decomposition to partition

[1]The simulator and datasets used in this paper are available to the community here: https://gitlab.com/onrg/multicloudsim.

the network into independent regions like cloud zones that can be simulated in parallel, with minimal synchronization needed only at boundary nodes. This hierarchical approach allows for localized testing of failures or traffic changes without full system recalculation, and the system supports both single-threaded mode (benefiting from caching alone) and multithreaded mode (leveraging parallel regional simulations) to significantly improve scalability and reduce runtime in large-scale multi-cloud environments.

In addition, *MCSim* supports hybrid emulation: users can export simulation-derived conditions to live hosts using Linux traffic control (tc), bridging simulation with testbed experimentation. This allows researchers to validate performance in virtualized environments using tools like iperf, netperf, or real cloud deployments, enhancing practical utility of *MCSim*.

To assess the fidelity and utility of *MCSim*, we conduct a multi-faceted evaluation grounded in real multi-cloud measurements. Our experiments validate that *MCSim* accurately reproduces performance trends observed in actual deployments, adapts correctly to parameter variations, and provides predictive insight into routing performance, bottlenecks, and failure recovery. We also demonstrate *MCSim*'s unique capabilities through detailed case studies, showing how it enables cost-performance analysis, failover simulations, and sensitivity testing at scales infeasible in real-world multi-cloud networks.

II. Related Work

Table I summarizes the landscape of network simulation tools and their multi-cloud capabilities. While each tool excels in its domain, none addresses the specific requirements of multi-cloud network analysis: cloud-native topology modeling, empirical trace integration, and cost-aware routing simulation.

Traditional Network Simulators. NS-3 [6] and OMNeT++ [7] provide the foundation for performance analysis by enabling detailed packet-level or modular simulation. These tools provide extensive support for protocol behavior and custom topologies, but lack built-in abstractions for multi-cloud environments. Specifically, they do not model cloud-specific constructs such as regions, availability zones, or economic costs. These capabilities are essential for accurate multi-cloud network modeling and are absent in existing simulators.

Data Center Network Simulators. Fabsim-X [9] target scalability at the data center level, supporting large endpoints and specialized workloads. However, they focus on tightly coupled infrastructures, without support for federated, heterogeneous, or geographically distributed multi-cloud networks. Similarly, Mininet [10] and GNS3 [11] offer realistic emulation using real networking stacks. These tools cannot model egress cost-driven multi-cloud environments where routing decisions must consider both performance and cost across cloud provider.

Cloud Resource Simulators. CloudSim [8] is one of the few simulators to explicitly support multiple cloud datacenters and VM scheduling, but it abstracts away the network layer and does not support inter-cloud routing, latency variability, and the complex interplay between network performance and egress costs that drive real-world multi-cloud decisions.

III. *MCSim* Overview

MCSim is a discrete-event network simulator designed to model and analyze multi-cloud environments under realistic and dynamic conditions. Implemented in Python and built on the SimPy framework [12], it provides a modular and extensible platform for simulating network topologies, application workloads, and infrastructure behavior across cloud providers such as AWS, Azure, and GCP. As shown in Figure 1, the simulator is organized around a set of core components that support realistic packet-based communication in heterogeneous, geo-distributed cloud deployments.

Hierarchical topology model. At the core of *MCSim* is a *hierarchical topology model*, where users define networks composed of cloud providers, each with multiple regions, zones, and virtual nodes. These nodes are connected by edges that capture performance (e.g., latency, bandwidth, jitter, packet loss) as well as cost (e.g., egress costs) metrics.

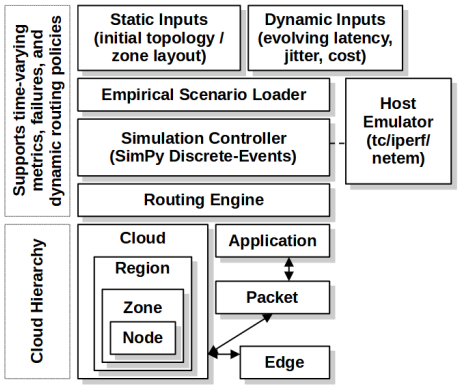


Fig. 1: *MCSim* architecture.

The topology and metrics can be specified manually as well as imported from empirical datasets or live measurements, enabling simulations that closely reflect the behavior of real-world multi-cloud networks. This flexibility allows researchers to model both static and time-varying conditions, including daily performance fluctuations, infrastructure outages, or pricing changes.

Pluggable routing engine. *MCSim* includes an out-of-band *pluggable routing engine* that supports a variety of algorithms, such as shortest-path-first (SPF) and overlay-based routing. Routing decisions can be made based on any available metric. Latency is used by default, but policies based on cost or reliability can also be applied. During simulation, packets are generated by applications running on nodes. These applications include tools such as ping, traceroute, and bulk UDP traffic generators. As packets traverse the network, they accumulate latency and experience potential loss according to the properties of the links they encounter. At the destination, applications can measure arrival times, jitter, or packet loss to characterize the end-to-end behavior of simulated flows.

Caching and Spatial Parallelization. As the number of overlay paths increases with the number of providers in a multi-cloud environment, a generic optimization formulation for tasks such as overlay-based routing becomes inefficient.

Simulator	Core Purpose	Multi-Cloud Support	Topology Modeling	Data Replay	Workload Modeling	Egress Cost Modeling
NS-3 [6]	Discrete-event Internet protocol simulation	✗	Custom topologies, no native cloud regions/zones	✗	Protocol-based traffic flows	✗
OMNeT++ [7]	Modular network simulation (wired/wireless)	✗	Highly extensible, no cloud abstractions	✗	Synthetic generators, INET models	✗
Fabsim-X [9]	Congestion control in large-scale DCNs	✗	Fat-tree DCN topologies only	✗	Congestion-focused traffic models	✗
Mininet [10]	Real Linux-based network emulation	✗	Custom topologies, no cloud modeling	✗	Real apps on virtualized hosts	✗
CloudSim [8]	Cloud resource and VM-level simulation	✓	High-level DC abstraction, no network detail	✗	VM-based tasks (Cloudlets)	✗
GNS3 [11]	Real-device network emulation	✗	Device-focused, no cloud or simulation support	✗	Real protocols and stacks	✗
***MCSim* (Ours)**	Multi-cloud topology and performance simulation	✓	Cloud-native hierarchy (region/zone/VM), cost-aware links	✓	Simulated + custom cloud application support	✓

TABLE I: Comparison of network simulators with respect to their multi-cloud network simulation capabilities.

In other words, simulating a generic solution that dynamically identifies routable paths to mitigate performance issues while being *optimal* w.r.t. diverse egress costs and varying network performance can significantly hinder the scalability of *MCSim*. To address this, *MCSim* introduces two key innovations.

First, *MCSim* leverages two forms of *caching* to reduce redundant route computations and repeated data access. *Route caching* stores and exports previously computed paths along with associated metrics such as latency and egress cost, avoiding the need to recalculate them on each query. In parallel, *snapshot caching* captures the state of the network at selected intervals, enabling quick rollback or reuse when similar simulation scenarios arise.

Second, since *MCSim* adopts a hierarchical topology model, it naturally enables the use of *spatial domain decomposition* [13] to accelerate routing simulations. This approach partitions the network topology into regional subgraphs, with each thread responsible for routing computations within a specific cloud region. Inter-region communications are handled through boundary node synchronization, reducing computational complexity from $O(n^2)$ for full-graph routing to $O(k \times (n/k)^2)$ where k is the number of regions. This approach leverages the natural locality of cloud network traffic, where most communications occur within regions or between adjacent regions.

MCSim supports both single-threaded and multithreaded execution modes. In single-threaded mode, caching alone improves efficiency by avoiding redundant computations, while multithreaded mode leverages spatial domain decomposition to run simulations of different network regions in parallel, further boosting scalability and reducing overall runtime.

Hybrid Mode. Beyond these modes, *MCSim* offers a *hybrid mode* for coordinated host emulation between local and cloud resources. The system can calculate, then export, simulated path characteristics such as latency, jitter, or bandwidth limits and apply them to a host machine using the Linux traffic control (tc) interface. This capability enables testbed nodes or virtual machines to experience network conditions identical to those observed in the simulation. As a result, users can run real benchmarking tools, such as iperf3 [14], HammerDB [15], etc. under network conditions synchronized with simulated workloads. This bridging between simulation and emulation is especially useful for deployment testing and validation.

Measurement Support. *MCSim* also includes built-in support for statistical data collection. Users can log key performance indicators, including latency distributions, jitter variance, loss rates, and total multi-cloud egress costs. These statistics make it possible to conduct validation studies, perform sensitivity analysis, profile routing behavior under stress, or simulate failure scenarios across heterogeneous cloud infrastructures.

IV. Experimental Setup

Cloud Measurement Setup. To evaluate the realism and fidelity of *MCSim*, we first collected empirical network performance data across the three major public cloud providers: Amazon Web Services (AWS), Google Cloud Platform (GCP), and Microsoft Azure. Our goal was to measure inter-region latency, jitter, and packet loss using virtual machines (VMs) deployed in every available global region of each provider.

Specifically, we deployed `t2.micro` or `t3.micro` instances on AWS across 24 regions, `e2-micro` instances on GCP across 41 regions, and `B1s` instances on Azure in 40 regions—totaling 105 VMs. Each VM was provisioned with at least 1 vCPU and 1 GB RAM and ran Ubuntu Linux 24.04.

Between January 31 and February 17, 2025, we conducted active measurements in 10-minute intervals. Every VM attempted to probe every other VM using both ICMP-based `ping` and TCP-based `ping` tools. For each probe, five packets were sent, yielding over 10,000 measurements per tool per interval, and generating over 50 million datapoints in total.

From this dataset, we derived per-link metrics such as round-trip time (RTT), jitter, and packet loss rate. These metrics were aggregated using median values and converted into topology files for use within *MCSim*, enabling realistic multi-cloud simulations based on real-world network behavior.

Simulation Testbed Setup. To evaluate different multi-cloud scenarios, we developed automation scripts for configuring and running simulations in *MCSim*. These scripts interact with the simulation controller to load empirical network data, configure topologies, and create application workflows.

The controller supports a range of test scenarios, from simple tools like ping and traceroute to more complex benchmarks such as bulk data transfers and database workloads (e.g., HammerDB [15]). Users can configure routing policies, propagation delays, and application-level behaviors. The simulator currently runs on a single CPU core, and during peak load, fully consumes one CPU core/thread.

Hybrid Deployment Setup. For hybrid experiments combining local and cloud resources, *MCSim* uses two sets of virtual machines: one hosted locally and one in the cloud. Locally, we used Oracle VirtualBox to run Ubuntu Server 24.04 VMs on a Windows 11 Pro machine equipped with 32 GB RAM and an Intel Core i7 CPU (8 cores / 16 threads @ 2.9 GHz). These VMs, provisioned with 2 vCPUs and 8 GB RAM each, were connected via a NAT network, with simulated latencies and metrics configured on each interface to emulate wide-area behavior. For the cloud side, *MCSim* deployed VMs in one region from each major cloud provider: a `t2.large` instance on AWS, an `e2-standard-2` instance on GCP, and a `Standard_D2s_v3` instance on Azure. Each instance had 2 vCPUs, 8 GB RAM, and ran Ubuntu 24.04.

This hybrid setup allows *MCSim* to test realistic distributed deployments, combining real inter-cloud dynamics with configurable local experimentation.

***MCSim* Parameters.** One-way latencies were derived as RTT/2, acknowledging this approximation's limitations for asymmetric paths. Jitter was estimated using the mean difference between RTT values per ping sequence. Packet loss estimation used $p = 1 - sqrt(1 - L)$ to convert round-trip loss L to one-way probability, based on independent loss assumptions. Jitter modeling employed bounded Gaussian distributions ($\pm 2\sigma$) to avoid unrealistic extreme values while preserving statistical characteristics.

V. Evaluation Results

Our evaluation focuses three research questions (RQs):

- **RQ1 (Fidelity)** (§ V-A): How accurately does MCSim reproduce real-world multi-cloud network behavior?
- **RQ2 (Scalability)** (§ V-B): How does MCSim's performance scale with network size and complexity?
- **RQ3 (Utility)** (§ V-C): Can MCSim provide actionable insights for multi-cloud network optimization and failure planning?

Our results demonstrates that *MCSim* can: simulate realistic multi-cloud network behavior, scale efficiently to large topologies, and offer actionable insights for cost-performance optimization and failure planning via what-if scenarios.

A. RQ1: Simulation Fidelity

To ensure that *MCSim* accurately replicates real-world multi-cloud behavior, we conduct three levels of validation.

Empirical Trace Validation. First, we feed *MCSim* with median latency, jitter, and packet loss traces collected from inter-cloud measurements across AWS, GCP, and Azure. These traces are replayed as time-varying link parameters during simulation. We then compare simulated end-to-end metrics against the observed empirical values.

We first present three illustrative comparisons between the real and simulated latency metrics gathered. `aws.us-west-2` (Oregon) to `gcp.us-east1` (South Carolina) is shown in Figure 2a. The distribution of real-world measurements approximates a Gaussian shape but deviates with occasional outliers. In contrast, the simulator uses an idealized Gaussian distribution, leading to more consistent and less variable results.

In Figure 2b, we see `azure.westus2` (Washington) to `aws.ca-central-1` (Montreal, Canada). This is an example of unpredictable variability in real-world cloud deployments that the simulator alone cannot reproduce accurately. To address such cases, *MCSim* includes a *replay mode*, which allows it to ingest and use real-world latency measurements collected from cloud networks.

Next, the comparison for `gcp.us-east1` (South Carolina) to `azure.westus3` (Arizona) shown in Figure 2c shows that the simulated metrics closely match the real measurements, with occasional variability (i.e., exhibiting heavier tail) in the real data.

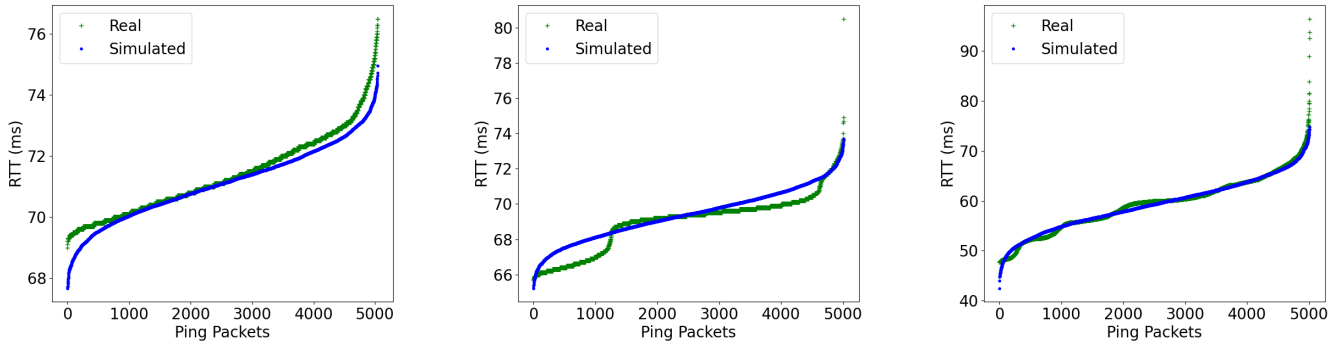
In all our experiments across several test regions in North America, *MCSim* achieved less than 1% error in average latency and closely mirrored the observed delay distributions, validating its ability to reproduce realistic inter-cloud behavior. While real measurements exhibit occasional outliers and asymmetric distributions, *MCSim*'s Gaussian modeling captures the central tendency accurately. The simulator's idealized distributions provide conservative estimates suitable for capacity planning, though they may occasionally underestimate tail latencies in production scenarios.

Statistical Accuracy. Second, we assess the accuracy of *MCSim* by comparing simulation outputs to application-level metrics gathered from real deployments. We analyzed metrics such as latency, response times, etc., using root mean square error (RMSE), mean absolute error (MAE), and correlation coefficients. Additionally, we employ the Kolmogorov–Smirnov (K-S) test to evaluate distribution similarity.

Figure 3a presents the sorted RMSE values between real and simulated median latency across region pairs. Average RMSE is approximately 2.4 ms for inter-cloud pairs and 1.5 ms for intra-cloud pairs. Notably, $\sim$87% of inter-cloud and $\sim$98% of intra-cloud region pairs have RMSE values below 5 ms, indicating alignment between simulated and real-world measurements. The higher RMSE values in a small fraction of inter-cloud pairs reflect the greater variability and unpredictability inherent to inter-cloud links.

Figure 3b shows the sorted Mean Absolute Error (MAE) values for the same set of region pairs. The results indicate that $\sim$92% of inter-cloud and $\sim$94% of intra-cloud pairs exhibit MAE values below 2 ms, suggesting that *MCSim* maintains a low average deviation from real latency values, further supporting its accuracy.

Figure 3c displays the Pearson Correlation Coefficients, capturing how well the trends in simulated latency match



(a) Oregon (AWS) to South Carolina (GCP). (b) Washington (Azure) to Montreal (AWS). (c) South Carolina (GCP) to Arizona (Azure).

Fig. 2: Comparison of real vs. simulated latency across different inter-cloud region pairs.

those in the real data. Approximately 95% of inter-cloud and 82% of intra-cloud region pairs show a correlation coefficient greater than 0.5, demonstrating that the simulator not only approximates the magnitude of latency but also closely tracks temporal variations in real-world behavior.

Distribution Similarity. Third, Kolmogorov-Smirnov (K-S) tests yield p-values of 1.0 across all region pairs, providing strong evidence that simulated and real distributions are statistically indistinguishable. (Resulting graphs is not shown due to space constraints). This validates MCSim's ability to reproduce not just average behavior but entire performance distributions.

Sensitivity Analysis. Last, we perform sensitivity analysis by varying one key input at a time (e.g., inter-region latency between GCP and Azure) in controlled increments. We then observe the resulting changes in application throughput, failover triggering, and routing decisions.

To evaluate the effects of loss on applications, we conduct an experiment where we first collect baseline measurements using HammerDB [15] between `gcp.us-south1` (Texas) and `azure.westus2` (Washington). With the baseline measurements, we duplicate those measurements using *MCSim*' host emulation interface. We then proceed to use *MCSim* to simulate loss rates from 0% to 50% in increments of 10 by taking the average of 10 scores.

In Figure 4 we present a comparison between real world and simulated New Orders Per Minute (NOPM) and Transactions Per Minute (TPM) benchmark values using loss rates from 0-50%. *MCSim* demonstrates as loss numbers increase in the real world and simulated measurements, application throughput drops at the same rate. This demonstrates that *MCSim* is able to accurately model and predict application performance.

B. RQ2: Scalability Analysis of MCSim

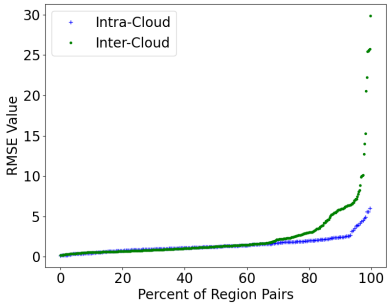
To evaluate scalability, we measure the runtime behavior of *MCSim* under increasing workload sizes and analyze its resource consumption.

Runtime Scaling. To demonstrate MCS's runtime scaling capabilities, we simulate topologies with increasing node counts from 50 to 1000 virtual endpoints. Each topology consists of 2 clouds with 25 regions per cloud, one availability zone per region, and a variable number of nodes per zone. We increase the total node count by 50 per zone. Network edges are configured with random round-trip time (RTT) and jitter values, while packet loss is set to zero for consistent performance measurement. Starting with a central node pair, we progressively add nodes to the topology, connecting each new node to two existing nodes via bidirectional edges to ensure full connectivity and reachability.

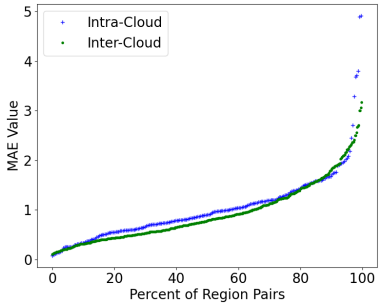
Each node executes a ping application that sends 10 ICMP echo requests to a randomly selected target node, generating 10 corresponding echo replies. This results in 1,000 total packets for the 50-node topology, scaling to 20,000 packets for the 1,000-node configuration. Our event throughput calculations focus specifically on ICMP echo requests and replies, though the simulator handles additional network behaviors including routing table lookups and packet forwarding. As expected for a discrete-event simulator, CPU utilization reaches 100% during execution.

Figure 5 shows stark differences in computational efficiency among the four approaches using a logarithmic scale. Single thread execution of *MCSim* with no caching or spatial parallelization shows exponential degradation in performance, beginning at approximately 6 seconds for 200 nodes and escalating dramatically to over 1500 seconds for 1000 nodes. This represents more than a 250x increase in execution time, highlighting the severe scalability limitations of sequential routing table processing for this type of computation. Multi-thread approach provides meaningful but limited improvement, reducing execution times by roughly 20-30% compared to single-threaded execution, yet still requiring over 1200 seconds for the larger number of nodes. This suggests that while parallelization helps, the underlying algorithm may have inherent bottlenecks that prevent effective utilization of multiple threads (as described in § III).

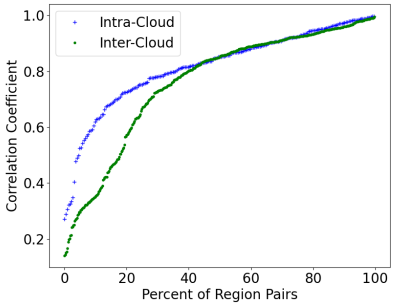
MCSim approaches deliver better performance improvements through caching (*MCSim* Cached) and spatial parallelization (*MCSim* Spatial). *MCSim* Spatial emerges as the clear performance leader, maintaining remarkably consistent execution times across all node sizes, never exceeding 20



(a) RMSE between real and simulated latency for intra- and inter-cloud measurements.



(b) MAE between real and simulated latency for intra- and inter-cloud measurements.



(c) Pearson correlation between real and simulated latency for intra- and inter-cloud measurements.

Fig. 3: Comparison of real and simulated latency metrics using RMSE, MAE, and Pearson correlation for both intra-cloud and inter-cloud scenarios.

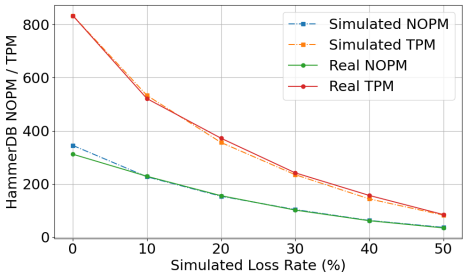


Fig. 4: Real and simulated HammerDB NOPM and TPM scores from Texas (GCP) to Washington (Azure).

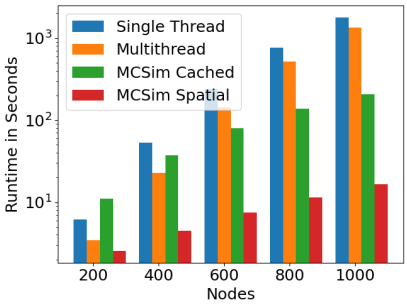


Fig. 5: Simulating total runtimes using a single thread, multiple thread routing calculations, cached topologies, and spatial parallel processing using cached topologies.

seconds even for 1000 nodes. This represents a performance improvement of nearly two orders of magnitude compared to single threaded execution. MCSim Cached shows better performance for larger numbers of nodes (e.g., completing 600-1000 node instances faster than Single Thread or Multithread) where spatial decomposition is impractical.

Nevertheless, the dramatic speedup due these approaches in *MCSim* enables large-scale multi-cloud experiments previously infeasible.

Memory Trade-offs. The memory consumption patterns in Figure 6 reveal the cost of these dramatic performance improvements. The graph shows memory consumption patterns of *MCSim*, with memory usage plotted on a logarithmic scale. Single thread and Multithread execution incur relatively modest and consistent memory usage that grows gradually with node size, reaching around 50 MB at 1000 nodes. In contrast, the MCSim optimizations show dramatically higher memory consumption. *MCSim* Cached exhibits particularly aggressive memory usage, scaling exponentially from about 70 MB at 200 nodes to over 1000 MB at 1000 nodes. *MCSim* Spatial also shows substantial memory requirements, though somewhat more controlled than the cached version, reaching approximately 400 MB for 1000 nodes.

We believe the memory increase is acceptable for modern systems and enables interactive simulation of realistic multi-cloud topologies.

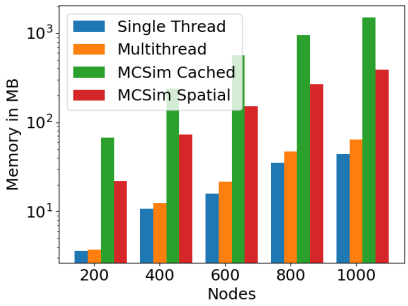


Fig. 6: Memory usage vs. the number of nodes.

Once routing tables are computed, *MCSim* can also optionally export them to external YAML files for reuse. When these *cached* tables are later imported, the simulator reconstructs the exact same routing state as the original calculation. Because the cached data mirrors the live routing results, fidelity remains identical whether routes are generated in real-time or loaded from a file.

Thus, scalability of the topology has no impact on simulation fidelity: that is, the correctness of results is preserved regardless of network size or caching strategy.

C. RQ3: Practical Utility of MCSim

Here, we present three case studies to demonstrate the practical utility of *MCSim*.

Case Study 1: Latency Reductions via Opportunistic Relay Node Selection. Within the North American region, multiple latency optimization opportunities exist across cloud providers, while cost optimization proves more challenging due to similar pricing structures between region pairs [1]. So, our goal is to simulate cost- and performance-aware routing calculations in multi-cloud environments by reproducing the results from Yeganeh et al. [1].

To this end, we used *MCSim* to quantify latency reductions achievable through relay node deployment and measure the corresponding cost increases. For each cloud provider pair, we evaluated every North American region as a potential relay point to determine whether routing through that region could reduce end-to-end latency between source and destination regions. We simultaneously tracked the egress cost increases required to achieve these latency optimizations. MCS successfully captured both latency savings and cost trade-offs. Figure 7 presents the percentage latency reduction achieved and identifies the cloud provider used for relay routing. Results show maximum latency reductions of approximately 27%, with an average egress cost increase of 57% across all inter-cloud paths utilizing single relay nodes that improved latency performance.

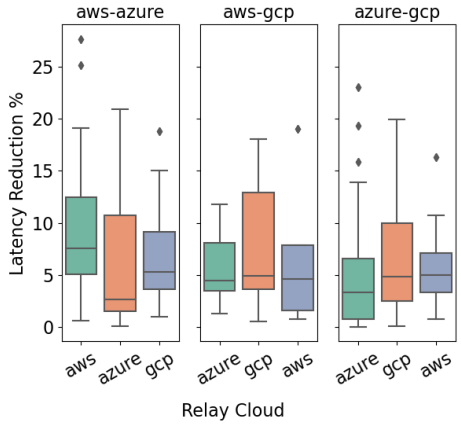


Fig. 7: Inter-cloud latency reductions in North America via relay nodes. Graph shows cloud pairs (top), the cloud of the relay node (x-axis), and latency reductions (y-axis).

We replicated Yeganeh et al.'s distribution of latency reduction ratio for inter-CP paths of each CP analysis of North American multi-cloud routing, validating MCSim's ability to reproduce published research. In addition, *MCSim*'s cost-aware routing enables systematic exploration of this trade-off space.

Case Study 2: Multi-Cloud Network Resilience. In a multi-cloud environment, resilience is one of the key promises as leveraging multiple providers should, in theory, protect applications from regional outages or single points of failure. But what really happens when an entire cloud region goes down (e.g., due to natural disasters [16])? How does traffic reroute, how are latency and costs impacted? Understanding these dynamics via *what-if scenarios* via simulations is crucial for designing robust multi-cloud strategies that can truly withstand real-world disruptions.

To evaluate *MCSim*'s failure response capabilities, we injected failures into simulated links and nodes (simulating regional outages) within the North American infrastructure encompassing all three major cloud providers. We implemented two callback functions to disable and subsequently re-enable specific regions. The first callback, triggered at simulation time 50,000, disabled the `gcp.us-south1` (Texas) and `gcp.us-central1` (Iowa) regions. The second callback, executed at simulation time 100,000, restored these regions. Each callback triggered routing table recalculation while the simulation clock remained paused. Additionally, we configured a ping application to generate 15,000 ICMP echo requests at 100ms intervals from `gcp.us-west4` (Las Vegas) to `azure.eastus` (Virginia), monitoring RTT and TTL values to calculate hop counts.

During the experiment, traffic continued flowing seamlessly when regions were disabled and routing tables recalculated, as demonstrated in Figure 8. However, traffic was rerouted through a path requiring one additional hop. The average RTT increased modestly from approximately 60ms to 63ms, indicating that an alternative path avoiding the disabled nodes was available with only a 3ms performance penalty.

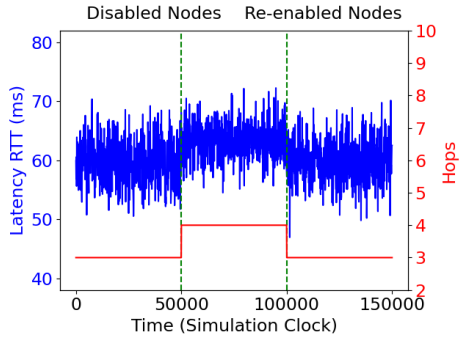


Fig. 8: Failure response to the disabling of two central regions when traffic moves from gcp.us-west4 (Las Vegas) to azure.eastus (Virginia).

Case Study 3: Hybrid Emulation of a Multi-Cloud Application. Finally, we explore the ability of *MCSim* to support hybrid testbed integration by exporting simulated network conditions into live virtualized environments using Linux tc. For this experiment, we deployed instances across all three major cloud providers: AWS (`us-east-1`), GCP (`us-south1`), and Azure (`westus2`). To ensure adequate computational resources, we provisioned cloud instances with 2 vCPUs and 8 GB RAM using `t2.large` (AWS), `e2-standard-2` (GCP), and `Standard_D2s_v3` (Azure) instance types.

We configured two Oracle VirtualBox VMs using *MCSim* interface features to match the network characteristics observed between cloud providers. Through experimentation, we determined that the host machine required approximately twice the RAM and more than double the vCPU resources of the target VMs to accurately simulate inter-cloud communication. Our host system utilized 32 GB RAM and 8 cores (16 vCPU threads) to meet these requirements.

To demonstrate *MCSim*'s hybrid mode, we applied *MCSim*-generated network parameters (latency, jitter, and packet loss) to virtual network interfaces on VMs running HammerDB, a real-world database benchmarking tool. We then compared the performance metrics (new operations per minute (NOPM) and transactions per minute (TPM)) between actual inter-cloud deployments and *MCSim* predictions using local VM simulation.

The results (shown in Figure 9) demonstrate remarkable fidelity between simulation and reality. As shown in the comparative analysis, HammerDB performance measurements reveal striking consistency across all three cloud provider pairs: (1) **AWS to GCP**: Both TPM and NOPM metrics show tight clustering around 1200 and 500 respectively, with simulated results precisely matching real-world measurements. (2) **GCP to Azure**: Performance metrics cluster around 850 TPM and 375 NOPM, again showing clear correlation between simulation and actual deployment. (3) **Azure to AWS**: Lower performance characteristics around 600 TPM and 250 NOPM are accurately reproduced in simulation.

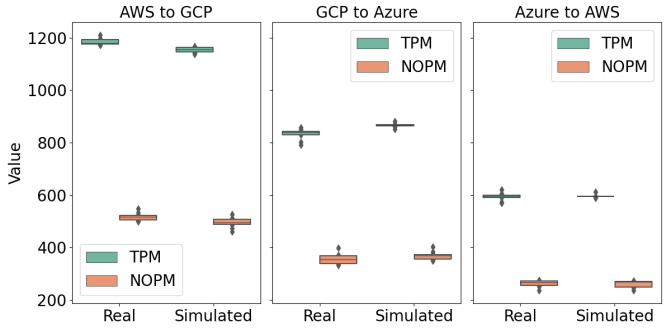


Fig. 9: Comparing the HammerDB benchmarking test results for real and simulated network configurations.

Across all test scenarios, the simulated measurements deviate by less than 2% from their real-world counterparts, with box plots showing nearly identical median values and variance patterns. This accuracy highlights *MCSim* practical utility: the results can reliably be projected into live testing environments, enabling developers to validate and fine-tune applications using local simulation before committing to expensive multi-cloud deployments.

VI. Summary & Future Work

Multi-cloud networking presents unique challenges that existing simulation tools cannot adequately address. *MCSim* bridges this gap by providing the first simulation framework designed specifically for empirical, cost-aware multi-cloud network analysis. Our comprehensive evaluation demonstrates high fidelity reproduction of real-world behavior, efficient scalability through spatial parallelization, and practical utility for cost-performance optimization and failure planning. *MCSim*'s hybrid simulation capability represents a significant advancement, enabling researchers to bridge simulation and real-world validation seamlessly. The empirical foundation of *MCSim* 50+ million measurements across major cloud providers ensures simulation results reflect actual cloud behavior rather than synthetic assumptions.

Future work includes advanced transport protocol modeling, real-time cloud billing integration [17], and support for emerging cloud-native architectures such as service mesh and edge computing.

Acknowledgments

We thank the anonymous reviewers for their feedback. This work is supported by the National Science Foundation (NSF) through CNS-2145813. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of NSF.

References

[1] B. Yeganeh, R. Durairajan, R. Rejaie, and W. Willinger, "A case for performance-and cost-aware multi-cloud overlays," in *2023 IEEE 16th International Conference on Cloud Computing (CLOUD)*. IEEE, 2023, pp. 560–566.

[2] Form3, "Form3 — Payment technology reimagined," 2025, accessed: 2025-06-01. [Online]. Available: https://www.form3.tech/

[3] Stake, "Stake — Australia-based Online Investing Platform," 2025, accessed: 2025-06-01. [Online]. Available: https://hellostake.com/au

[4] C. McAllister, "Multi-cloud architecture: Three real-world examples from fintech," September 2023, accessed: 2025-06-01. [Online]. Available: https://www.cockroachlabs.com/blog/fintech-multi-cloud-architecture/

[5] J. Colton, R. Durairajan, R. Rejaie, and W. Willinger, "On the Impact of Submarine Cable Deployments on Multi-Cloud Network Latencies," in *2024 IEEE 13th International Conference on Cloud Networking (CloudNet)*. IEEE, 2024, pp. 1–9.

[6] ns-3 project, "ns-3: Network Simulator 3," https://www.nsnam.org, 2025, accessed: 2025-05-31.

[7] OMNeT++ Community, "OMNeT++ Discrete Event Simulator," https://omnetpp.org, 2025, accessed: 2025-05-31.

[8] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose, and R. Buyya, "CloudSim: A Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms," *Software: Practice and Experience*, vol. 41, no. 1, pp. 23–50, 2011.

[9] M. Musleh, R. Penaranda, A. Alemania, P. Y. Segura, G. Wu, J. Zieliński, K. Raszkowski, N. Ni, S. Diesing, A. Kurpad, R. Huggahalli, C. E. Bruns, S. Miller, and S. Sen, "Fabsim-X: A simulation framework for the analysis of large-scale topologies and congestion control protocols in data center networks," in *Proceedings of the 2020 28th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*. IEEE, 2020, pp. 1–8.

[10] B. Lantz, B. Heller, and N. McKeown, "Mininet: Rapid Prototyping for Software Defined Networks," http://mininet.org, 2025, accessed: 2025-05-31.

[11] GNS3 Technologies Inc., "GNS3: Graphical Network Simulator," https://www.gns3.com, 2025, accessed: 2025-05-31.

[12] Team SimPy, "Simpy: Discrete event simulation for python," https://simpy.readthedocs.io, 2025, accessed: 2025-05-31.

[13] A. P. Fitzgerald, B. Kochunas, S. Stimpson, and T. Downar, "Spatial Decomposition of Structured Grids for Nuclear Reactor Simulations," *Annals of Nuclear Energy*, vol. 132, pp. 686–701, 2019.

[14] E. S. N. (ESnet), "iperf3: A TCP, UDP, and SCTP Network Bandwidth Measurement Tool," https://software.es.net/iperf/, 2014, accessed: 2025-06-01.

[15] S. Shaw, "HammerDB: Database Load Testing and Benchmarking Tool," https://www.hammerdb.com/, 2025, accessed: 2025-06-01.

[16] J. Mayer, V. Sahakian, E. Hooft, D. Toomey, and R. Durairajan, "On the Resilience of Internet Infrastructures in Pacific Northwest to Earthquakes," in *Passive and Active Measurement, Virtual Event*, 2021.

[17] Google Cloud, "Google cloud billing cost estimation api (rest)," https://cloud.google.com/billing/docs/reference/cost-estimation/rest, 2025, accessed: 2025-05-31.